XML-Scripting mit XSLT für SysAdmins

Gerd Aschemann

01.03.2002

GUUG FFG 2002, Bochum

Konventionelle Konfigurationsdateien

```
...
test.cno:testacc
home.test.cno:testacc
...
```

```
<VirtualHost www.test.cno>
ServerAdmin testacc@test.cno
ServerName www.test.cno

DocumentRoot /web/test.cno/web/www/htdocs
...
```

```
$TTL    1D
@       IN      SOA     ns0.test.cno. hostmaster.test.cno. (
                        2002022704      ; Serial
                        8H              ; Refresh
                        2H              ; Retry
...
          MX 20   mailbackup.test.cno.
...
test.cno.               A       10.13.08.15
mailbackup              A       10.08.15.13
```

Zukünftige Konfigurationsdatei(en)

```
<zone>
  <domain>test.cno</domain>
  <user><short>ta</short>testacc</user>
...
  <mx>
    <prio>20</prio>
    <host>mailbackup</host>
  </mx>
...
  <a>
    <host>test.cno.</host>
    <ip>10.13.08.15</ip>
  </a>
...
  <cname http='true()'>
    <name>www</name>
    <host>home</host>
  </cname>
```

```
...
  <a>
    <host likedomain=true()/>
    <ip>10.13.08.15</ip>
  </a>
...
```


Motivation

- Endziel: Viele konventionelle Konfigurationsdateien mit vielen verschiedenen Formaten und redundanten Informationen durch eine einzige Konfigurationsdatei ersetzen.
- Zwischenschritt: Aus einer Quelle verschiedene Ziele (mit gleichen Informationen) generieren
- Redundanz (mehrfacher Pflegeaufwand) vermeiden
- dadurch Fehlerhäufigkeit verringern
- Informationen für zukünftige weitere Verarbeitungsschritte, z.B. Dokumentation, bereitstellen.

Realisierung: XML + XSL(T), Beispiel 1: `virtualdomains`

```
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="text" encoding="iso-8859-1" indent="no"/>

  <xsl:template match="zone">
    <xsl:value-of select="domain"/>:<xsl:value-of select='user/text()'/>
    <!-- Provide an empty line in the output ... -->
    <xsl:text>
    </xsl:text>

    <xsl:apply-templates select="a"/>
    <xsl:apply-templates select="cname"/>

  </xsl:template>
```

```
<xsl:template match="a">
<xsl:if
  test="substring(host/.,string-length(host/.)) != '.'">
  <xsl:value-of select='host'/>.
    <xsl:value-of select='../domain'/>:
    <xsl:value-of select='../user/text()'/><xsl:text></xsl:text>
</xsl:if>
</xsl:template>
```


Beispiel 2: `httpd.conf`

```
<xsl:stylesheet
  version="1.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transf
  xmlns:saxon="http://icl.com/saxon"
  saxon:trace="no"
  extension-element-prefixes="saxon">

  <xsl:import href="../site/site-defs.xsl"/>

  <xsl:template match="zone">
    <xsl:apply-templates select="cname[@http]"/>
    <xsl:apply-templates select="a[@http]"/>
  </xsl:template>
```

Verwendung von Saxon wg. bestimmter Eigenschaften aus XSL 2, s.u.

Modularisierung ...

Beispiel 2: `httpd.conf (2)`

```
<xsl:template match="zone/cname[@http]">

  <xsl:variable name="servername">
    <xsl:value-of select="name"/>.<xsl:value-of select="/zone/domain"/>
  </xsl:variable>

  <saxon:output method="text"
    href="httpdinc/{$servername}.inc"
    encoding="iso-8859-1">

    <xsl:call-template name="virtualhost"/>
  </saxon:output>
 </xsl:template>
```

Saxon: Ausgabe in Datei, in XSL 2.0? `<xsl:document ...`

Funktionen (ggf. mit Parametern).

Beispiel 2: `httpd.conf (3)`

```
<xsl:template name="virtualhost">
&lt;VirtualHost <xsl:value-of select='name'/>.<xsl:value-of select='/zone/domain'/>&gt;
ServerAdmin <xsl:value-of select='/zone/user'/>@<xsl:value-of select='/zone/domain'/>
ServerName <xsl:value-of select='name'/>.<xsl:value-of select='/zone/domain'/>

DocumentRoot <xsl:call-template name="webserver-rootdir"/>/htdocs
ErrorLog <xsl:call-template name="webserver-rootdir"/>/logs/error_log
TransferLog <xsl:call-template name="webserver-rootdir"/>/logs/access_log

&lt;Directory /&gt;
    Options -FollowSymLinks
    AllowOverride None
&lt;/Directory&gt;
```

Beispiel 3: dns.zone

```
 <xsl:template match="zone">
; THIS FILE IS GENERATED AUTOMATICALLY; DO NOT EDIT!!!
; (XSL: $Id: xml2dnszone.xsl,v 1.11 2002/01/30 23:31:51 ascheman Exp $)
;
; <xsl:value-of select='domain'/> zone lookup
;
; <xsl:value-of select='revision'/> / <xsl:value-of select='date'/>

$TTL            1D
@               IN      SOA     <xsl:call-template name="soa-origin"/> hostmaster.
  <xsl:value-of select='domain'/> . (
                                  <xsl:value-of select="$serial"/>        ; Serial
                                  8H              ; Refresh
                                  2H              ; Retry
                                  1W              ; Expire
                                  1D)             ; Minimum TTL
<xsl:text>
;
```

Woher kommt die Seriennummer?

Lösung: Externe Berechnung ... Kein XSL(T)/XML ... sondern perl ...

Zusammenfassung

- Pro:
  - Single Source
  - Keine Redundanzen
  - Regelbasierte Templatebeschreibung
  - Modularisierung/Portierbarkeit durch Auslagerung Site-spezifischer Anteile
  - Leichte (?) Erweiterbarkeit durch Anpassung des Schemas

- Contra:
  - Unübersichtliche Syntax, insb. bei XML-artigen Zieldokumenten
  - Keine (komplexen) Berechnungen
  - Verteilung des Modells auf mehrere Eingabedateien?

Informationsmodell, aber kein Kommunikationsmodell und kein fixes Berechenbarkeitsmodell!!!

Ausblick

- Zukünftig nur noch eine einzige Konfigurationsdatei
- Beschränkung auf eine Syntax (Benutzer + Werkzeuge)
- Einfache Weiterverarbeitung der Informationen
- Transformation und Kommunikation der Informationen zu beliebigen Zielen ...